Tally NP Sets and Easy Census Functions

Judy Goldsmith [1]
Department of Computer Science
University of Kentucky
Lexington, KY 40506, USA
`goldsmit@cs.engr.uky.edu`

Mitsunori Ogihara [2]
Department of Computer Science
University of Rochester
Rochester, NY 14627, USA
`ogihara@cs.rochester.edu`

Jörg Rothe [3]
Institut für Informatik
Friedrich-Schiller-Universität Jena
07740 Jena, Germany
`rothe@informatik.uni-jena.de`

March 19, 1998

Abstract

We study the question of whether every P set has an easy (i.e., polynomial-time computable) census function. We characterize this question in terms of unlikely collapses of language and function classes such as $\#P_1 \subseteq FP$, where $\#P_1$ is the class of functions that count the witnesses for tally NP sets. We prove that every $\#P_1^{PH}$ function can be computed in $FP^{\#P_1^{\#P_1}}$. Consequently, every P set has an easy census function if and only if every set in the polynomial hierarchy does. We show that the assumption $\#P_1 \subseteq FP$ implies $P = BPP$ and $PH \subseteq MOD_k P$ for each $k \geq 2$, which provides further evidence that not all sets in P have an easy census function. We also relate a set's property of having an easy census function to other well-studied properties of sets, such as rankability and scalability (the closure of the rankable sets under P-isomorphisms). Finally, we prove that it is no more likely that the census function of any set in P can be approximated (more precisely, can be n^α-enumerated in time n^β for fixed α and β) than that it can be precisely computed in polynomial time.

1 Introduction

Does every P set have an easy (i.e., polynomial-time computable) census function? Many important properties similar to this one were studied during the past decades to gain more insight into the nature of feasible computation. Among the questions that were previously studied are the question of whether or not every P set has an easy to compute ranking function [GS91,HR90], whether every P set is P-isomorphic to some rankable set [GH96], whether every sparse set in P is P-printable [HY84,AR88,RRW94], whether every infinite set in P has an infinite P-printable subset [AR88,HRW97a], whether every P-printable set is P-isomorphic to some tally set in P [AR88], and whether every P set admits easy certificate schemes [HRW97a,HRW97b], to name just a few. Some of those questions arise in the field of data compression and are related to Kolmogorov complexity, some are linked to the question of whether one-way functions exist.

Extending this line of research, the present paper studies the complexity of computing the census functions of sets in P. Census functions have proven to be a particularly important and useful notion in complexity theory, and their use has had a profound impact upon almost every area of the field. In particular, this regards the extensive literature related to the isomorphism conjecture of Berman and Hartmanis (e.g., [BH77,Mah82], and many other papers), the work on the existence of Turing-hard sparse sets (or of polynomial-size circuits) for various complexity classes (e.g., [KL80,KS85,BBS86,HR97]), the results relating the computation times for NP sets to their densities and the results on P-printability [HY84, AR88,RRW94,GH96], the upward separation technique (e.g., [Har83,HIS85,All91,RRW94, HJ95], see [HHH] for more recent advances that are not based on census functions), the results on positive relativization and relativization to sparse oracles (e.g., [Lon85,LS86,

BBS86]), the unexpected collapse of the strong exponential-time hierarchy [Hem89], and applications to extended lowness [HJRW].

Valiant, in his seminal papers [Val79a,Val79b], introduced #P, the class of functions that count the solutions of NP problems, and its tally version $\#P_1$ for which the inputs are given in unary. Although $\#P_1$ has not become as prominent as #P, it contains a number of quite interesting and important problems such as the problem `Self-Avoiding Walk` (see [Wel93]): Given an integer n in unary, compute the number of self-avoiding walks on the square lattice having length n and rooted at the origin. `Self-Avoiding Walk` is a well-known classical problem of statistical physics and polymer chemistry, and it is an intriguing open question whether `Self-Avoiding Walk` is $\#P_1$-complete (see [Wel93]). Known problems complete for $\#P_1$ [Val79b] have the form: Given an integer n in unary, compute the number of graphs having n vertices and satisfying a fixed graph property π.

In Section 3, we will characterize the question of whether every P set has an easy census function in terms of collapses of language and function classes that are considered to be unlikely. In particular, every P set has an easy census function if and only if $\#P_1 \subseteq FP$. The main technical contribution in Section 3 is Theorem 3.7: $\#P_1^{PH}$ is contained in $FP^{\#P_1^{\#P_1}}$. An immediate consequence of this result are upward collapse results of the form: the collapse $\#_1 \cdot P \subseteq FP$ implies the collapse $\#_1 \cdot PH \subseteq FP$. Thus, every P set has an easy census function if and only if every set in the polynomial hierarchy has an easy census function. Note that the corresponding upward collapse for the # operator applied to the levels of PH follows immediately from the upward collapse property of the polynomial hierarchy itself: $\# \cdot P \subseteq FP$ implies NP = P and thus PH = P; so, $\# \cdot PH = \# \cdot P \subseteq FP$. However, for the $\#_1$ operator this is not so clear, since the assumption $\#_1 \cdot P \subseteq FP$ merely implies that all *tally* NP sets are in P (equivalently, NE = E), from which one cannot immediately conclude that $\#_1 \cdot NP$ or even $\#_1 \cdot PH$ is contained in FP. In fact, Hartmanis, Immerman, and Sewelson [HIS85] show that in some relativized world, NE = E and yet the (weak) exponential-time hierarchy does not collapse. In light of this result, it is quite possible that the assumption of all tally NP sets being in P does not force all tally sets from higher levels of the polynomial hierarchy into P.

We also show that the assumption $\#P_1 \subseteq FP$ implies both P = BPP and $PH \subseteq MOD_kP$ for each $k \geq 2$ (Theorem 3.6), which provides further evidence that not all sets in P have a census function computable in polynomial time. We also relate a set's property of having an easy census function to other well-studied properties of sets, such as rankability [GS91] and scalability [GH96]. In particular, though each rankable set has an easy census function, we show that (even when restricted to the sets in P) the converse is not true unless P = PP. This expands the result of Hemaspaandra and Rudich that every P set is rankable if and only if P = PP [HR90] by showing that P = PP is already implied by the apparently weaker hypothesis that every P set *with an easy census function* is rankable.

Cai and Hemaspaandra [CH89] introduced the notion of enumerative counting as a way of approximating the value of a #P function deterministically in polynomial time.

Hemaspaandra and Rudich [HR90] show that every P set is k-enumeratively rankable for some fixed k in polynomial time if and only if $\#\text{P} = \text{FP}$. They conclude that it is no more likely that one can enumeratively rank all sets in P than that one can exactly compute their ranking functions in polynomial time. In Section 4, we similarly characterize the question of whether the census function of all P sets is n^α-enumerable in time n^β for fixed constants α and β, or equivalently, whether every $\#\text{P}_1$ function is n^α-enumerable in time n^β. We show that this hypothesis implies $\#\text{P}_1 \subseteq \text{FP}$, and we thus conclude that it is no more likely that one can n^α-enumerate the census function of every P set in time n^β than that one can precisely compute its census function in polynomial time.

Finally, Section 5 provides a number of relativization results.

2   Notation and Definitions

Fix the alphabet $\Sigma = \{0, 1\}$. Σ^* denotes the set of all strings over Σ, and $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$, where ϵ denotes the empty string. For any string $x \in \Sigma^*$, we denote the length of x by $|x|$. For any set $L \subseteq \Sigma^*$, the number of strings in L is denoted $|L|$, and the complement of L in Σ^* is denoted $\overline{L}$. Let $L^{=n}$ (respectively, $L^{\leq n}$) denote the set of strings in L of length n (respectively, of length at most n). As a shorthand, we use Σ^n to denote $(\Sigma^*)^{=n}$. For any set L, the *census function of L*, $census_L : \Sigma^* \to \mathbb{N}$, is defined by $census_L(1^n) \stackrel{\text{df}}{=} |L^{=n}|$,[1] and χ_L denotes the *characteristic function of L*, i.e., $\chi_L(x) = 1$ if $x \in L$, and $\chi_L(x) = 0$ if $x \notin L$. A set S is said to be *sparse* if there is a polynomial p such that for each length n, $census_S(1^n) \leq p(n)$. A set T is said to be *tally* if $T \subseteq \{1\}^*$. To encode pairs of strings, we use a one-one, onto pairing function, $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$, that is computable and invertible in polynomial time; this pairing function is extended to encode m-tuples of strings as is standard. For convenience, we will sometimes write m-tuples of strings $x_1, x_2, \dots, x_m \in \Sigma^*$ explicitly as $x_1 \# x_2 \# \dots \# x_m$, using a special separating symbol $\#$ not in Σ. We let $\leq$ denote the standard lexicographic order on Σ^*.

The definition of Turing machines and their languages, Turing transducers and the functions they compute, relativized (i.e., oracle) computations, (relativized) complexity classes, etc. is standard in the literature (see, e.g., the textbooks [HU79,BC93,Pap94]). We briefly recall the complexity classes most important in this paper. FP denotes the class of polynomial-time computable functions. FP_1 is the class of functions computable in polynomial time by deterministic transducers with a unary input alphabet. FE is the class of functions that can be computed by deterministic transducers running in time 2^{cn} for some constant c. Let $\text{E} \stackrel{\text{df}}{=} \bigcup_{c>0} \text{DTIME}[2^{cn}]$ and $\text{NE} \stackrel{\text{df}}{=} \bigcup_{c>0} \text{NTIME}[2^{cn}]$. An unambiguous Turing machine is a nondeterministic Turing machine that on each input has at most one

---

[1] The census function of L at n is often defined as the number of elements in L of length up to n in the literature. This definition and our definition are compatible as long as our computability admits subtraction. We also note that we let $census_L$ map strings 1^n (as opposed to numbers n in binary notation) to $|L^{=n}|$ to emphasize that the input to the transducer computing $census_L$ is given in unary.

accepting path. UP [Val76] (respectively, UE) is the class of all languages accepted by some unambiguous Turing machine running in polynomial time (respectively, in time 2^{cn} for some constant c).

For any nondeterministic Turing machine M and any input $x \in \Sigma^*$, let $\mathrm{acc}_M(x)$ denote the number of accepting paths of $M(x)$. A *spanP machine* [KST89] is an NP machine that has a special output device on which some output is printed for each accepting path. For any spanP machine M and any input $x \in \Sigma^*$, $\mathrm{span}_M(x)$ is defined to be the number of different outputs of $M(x)$ if $M(x)$ has at least one accepting path, and 0 otherwise. A *tally NP machine* (respectively, a *tally spanP machine*) is an NP (respectively, a spanP) machine with a unary input alphabet.

Definition 2.1   *1.* [Val79a,Val79b] $\#P \stackrel{\mathrm{df}}{=} \{\mathrm{acc}_M \mid M \text{ is an NP machine}\}$.

   2. [Val79b] $\#P_1 \stackrel{\mathrm{df}}{=} \{\mathrm{acc}_M \mid M \text{ is a tally NP machine}\}$.

   3. [KST89] $\mathrm{spanP} \stackrel{\mathrm{df}}{=} \{\mathrm{span}_M \mid M \text{ is a spanP machine}\}$.

   4. $\mathrm{spanP}_1 \stackrel{\mathrm{df}}{=} \{\mathrm{span}_M \mid M \text{ is a tally spanP machine}\}$.

   5. $\#E \stackrel{\mathrm{df}}{=} \{\mathrm{acc}_M \mid M \text{ is an NE machine}\}$.

   6. [MS72,Sto77]   *The polynomial hierarchy is inductively defined as follows:* $\Sigma_0^p \stackrel{\mathrm{df}}{=} P$, $\Sigma_k^p \stackrel{\mathrm{df}}{=} \mathrm{NP}^{\Sigma_{k-1}^p}$ *for* $k \geq 1$, *and* $\mathrm{PH} \stackrel{\mathrm{df}}{=} \bigcup_{i \geq 0} \Sigma_i^p$.

   7. [Gil77]   PP *is the class of languages L for which there exist a set A in P and a polynomial p such that for all strings $x \in \Sigma^*$,*

$$x \in L \iff |\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| \geq 2^{p(|x|)-1}.$$

   8. [Gil77]   BPP *is the class of languages L for which there exist a set A in P and a polynomial p such that for all strings $x \in \Sigma^*$,*

$$x \in L \implies |\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \notin A\}| \leq 2^{p(|x|)-2}, \text{ and}$$
$$x \notin L \implies |\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| \leq 2^{p(|x|)-2}.$$

   9. [CH90,Her90,BG92]   *For any fixed $k \geq 2$,* $\mathrm{MOD}_k\mathrm{P}$ *is the class of languages L for which there exist a set A in P and a polynomial p such that for all strings $x \in \Sigma^*$,*

$$x \in L \iff |\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| \not\equiv 0 \mod k.$$

   If $k = 2$, we write $\oplus P$ (introduced in [PZ83,GP86]) instead of $\mathrm{MOD}_2\mathrm{P}$.

   10. [OH93,FFK94]   SPP *is the class of languages L for which there exist a set A in P and a polynomial p such that for all strings $x \in \Sigma^*$,*

$$x \in L \implies |\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| = 2^{p(|x|)-1} + 1, \text{ and}$$
$$x \notin L \implies |\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| = 2^{p(|x|)-1}.$$

11. [KL80]   *For any language class $\mathcal{C}$, let $\mathcal{C}/\mathrm{poly}$ be the class of all languages L for which there exist a set $A \in \mathcal{C}$, a polynomial p, and an advice function $h : \Sigma^* \to \Sigma^*$ such that for each length n, $|h(1^n)| = p(n)$, and for every $x \in \Sigma^*$, $x \in L$ if and only if $\langle x, h(1^{|x|}) \rangle \in A$. For any function class $\mathcal{F}$, let $\mathcal{F}/\mathrm{poly}$ be the class of all functions g for which there exist a function $f \in \mathcal{F}$, a polynomial p, and an advice function $h : \Sigma^* \to \Sigma^*$ such that for each length n, $|h(1^n)| = p(n)$, and for every $x \in \Sigma^*$, $g(x) = f(\langle x, h(1^{|x|}) \rangle)$.*

We will use the common operator notation at times in order to generalize function classes such as #P and $\#\mathrm{P}_1$.

Definition 2.2 *For any language class $\mathcal{C}$, define*

1. *$\# \cdot \mathcal{C}$ to be the class of functions $f : \Sigma^* \to \mathbb{N}$ for which there exist a set $A \in \mathcal{C}$ and a polynomial p such that for each $x \in \Sigma^*$,*

$$f(x) = |\, \{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\} \,|, \text{ and}$$

2. *$\#_1 \cdot \mathcal{C}$ to be the class of functions $f : \Sigma^* \to \mathbb{N}$ for which there exist a set $A \in \mathcal{C}$ and a polynomial p such that for each $n \in \mathbb{N}$,*

$$f(1^n) = |\, \{y \mid |y| = p(n) \text{ and } \langle 1^n, y \rangle \in A\} \,|.$$

Definition 2.3   1. *A bijection $\phi : \Sigma^* \to \Sigma^*$ is a P-isomorphism if ϕ is computable and invertible in polynomial time.*

2. *A P-isomorphism ϕ is length-preserving if for all $x \in \Sigma^*$, $|\phi(x)| = |x|$.*

3. *A P-isomorphism ϕ mapping set $A \subseteq \Sigma^*$ to set $B \subseteq \Sigma^*$ is order-preserving if for any two strings x and y satisfying either $x, y \in A$ or $x, y \notin A$, if $x \leq y$, then $\phi(x) \leq \phi(y)$.*

Definition 2.4 [GS91] *The ranking function of a language $A \subseteq \Sigma^*$ is the function $r : \Sigma^* \to \mathbb{N}$ that maps each $x \in \Sigma^*$ to $|\, \{y \leq x \mid y \in A\} \,|$. A language A is rankable if its ranking function is computable in polynomial time.*

Goldsmith and Homer [GH96] introduced the property of scalability, a more flexible notion than rankability in which the rank of some given element within the set is not necessarily determined with respect to the lexicographic order of Σ^*, but rather with respect to *any* well-ordering of Σ^* that can be "scaled" by a polynomial-time computable and polynomial-time invertible bijection between $\mathbb{N}$ and Σ^*. Equivalently, the scalable sets are precisely those that are P-isomorphic to some rankable set. The definition below is based on this characterization.

Definition 2.5 [GH96] *A language A is scalable if it is P-isomorphic to a rankable set. For any oracle X, the X-scalable sets are those that are P^X-isomorphic to some set rankable in FP^X.*

3   Does P Have Easy Census Functions?

We start with exploring the relationships between the properties of a set being rankable, being scalable, and having an easy census function. Let A be any set (not necessarily in P). Consider the following conditions:

(i) A is rankable.

(ii) A has an easy census function.

(iii) A is P-isomorphic to some rankable set (i.e., A is scalable).

(iv) A is P-isomorphic to some rankable set via some length-preserving isomorphism.

(v) A is P-isomorphic to some rankable set via some order-preserving isomorphism.

It is immediately clear that for any set A, (i) implies each of (ii), (iv), and (v), and each of (iv) and (v) implies (iii). The next proposition shows that the rankable sets are closed under order-preserving P-isomorphisms (thus, conditions (i) and (v) in fact are equivalent) and that the class of sets having an easy census function is closed under length-preserving P-isomorphisms. The latter fact immediately gives that (iv) implies (ii), since each rankable set has an easy census function. The inclusion structure of the sets in P satisfying Properties (i) through (iv) is given in Figure 1.

Proposition 3.1   *1. The class of all rankable sets is closed under order-preserving P-isomorphisms.*

2. The class of sets having an FP-computable census function is closed under length-preserving P-isomorphisms.

Proof.   (1).   Let A be P-isomorphic to a rankable set B via some order-preserving isomorphism. Since B is rankable, $\overline{B}$ is rankable. Let respectively r and $\bar{r}$ be the ranking functions for B and $\overline{B}$. For any string $x \in \Sigma^*$, let $\mathrm{lex}(x)$ denote the lexicographic order of x. Define the function

$$r'(x) \stackrel{\mathrm{df}}{=} \begin{cases} r(x) & \text{if } x \in A \\ \mathrm{lex}(x) - \bar{r}(x) & \text{if } x \notin A. \end{cases}$$

Clearly, r' is computable in polynomial time and r' is the ranking function for A.

(2).   Let A be P-isomorphic to a set B with $census_B \in \mathrm{FP}$ via some length-preserving isomorphism ϕ. Then, $\phi(A^{=n}) = B^{=n}$. So, for every n, $census_A = census_B$. This implies $census_B \in \mathrm{FP}$. ∎

So we are left with only the four conditions (i) to (iv). Since there are nonrecursive sets with an FP-computable census function, but any set satisfying one of (i), (iii), or (iv) is in P, condition (ii) in general cannot imply any of the other three conditions. On the other hand, when we restrict our attention to the sets in P having easy census functions, we can
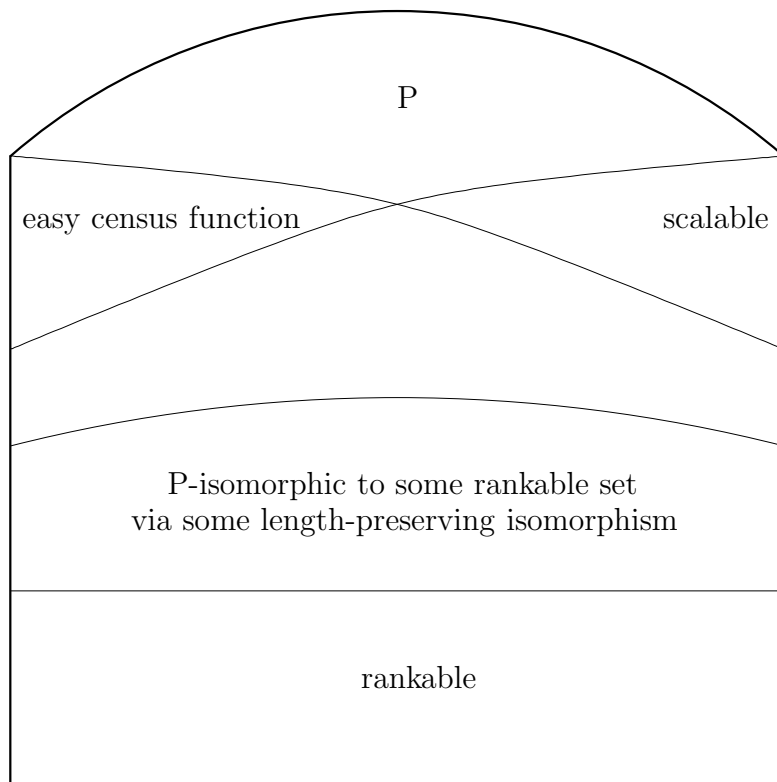
Figure 1: Inclusion structure of the sets in P satisfying Properties (i) through (iv).

show that (ii) implies (i) if and only if P = PP. Thus, even when restricted to P sets, it is unlikely that (ii) is equivalent to (i).

Theorem 3.2 *All* P *sets with an easy census function are rankable if and only if* P = PP.

Proof. Hemaspaandra and Rudich show that P = PP (which is equivalent to $P^{\#P} = P$) implies that every P set is rankable [HR90]. Conversely, let L be any set in PP, and let A be a set in P and p be a polynomial such that for all $x \in \Sigma^*$,

$$x \in L \iff |\{y \mid |y| = p(|x|) \text{ and } x\#y \in A\}| \geq 2^{p(|x|)-1}.$$

Define

$$T \stackrel{\text{df}}{=} \{b\#x\#y \mid x, y \in \Sigma^*, \ |y| = p(|x|), \ b \in \{0,1\}, \text{ and } \chi_A(x\#y) = b\}.$$

Clearly, $T \in$ P. Also, the census function of T is easy to compute: Given n in unary, compute the largest integer i such that $i + p(i) + 3 \leq n$. Then,

$$census_T(1^n) = \begin{cases} 2^{i+p(i)} & \text{if } i + p(i) + 3 = n \\ 0 & \text{if } i + p(i) + 3 < n. \end{cases}$$

Since $T \in$ P and $census_T \in$ FP, by hypothesis T is rankable. Let r be the ranking function for T. Since for each $x \in \Sigma^+$,

$$x \in L \iff r(0\#x\#1^{p(|x|)}) - r(1\#\widehat{x}\#1^{p(|\widehat{x}|)}) < 2^{p(|x|)-1},$$

where $\widehat{x}$ is the lexicographic predecessor of x, and since the predicate on the right-hand side of the above equivalence can be decided in polynomial time, it follows that $L \in$ P. ∎

Corollary 3.3 *All* P *sets are rankable if and only if all sets in* P *with an easy census function are rankable.*

One might ask whether or not all P sets outright have an easy census function (which, if true, would make Corollary 3.3 trivial). The following characterization of this question in terms of unlikely collapses of certain function and language classes suggests that this probably is not true. Thus, Corollary 3.3 is nontrivial with the same certainty with which we believe that for instance not all $\#P_1$ functions are in FP.[2]

Theorem 3.4 *The following are equivalent.*

1. *Every* P *set has an* FP-*computable census function.*

2. $\#P_1 \subseteq$ FP.

---

[2]It is not difficult to construct—by standard techniques—an oracle relative to which $\#P_1 \not\subseteq$ FP. On the other hand, we will show in Section 5 that, relative to some oracle, $\#P_1 \subseteq$ FP, yet $\#P \neq$ FP (and thus PP $\neq$ P).

3. $\#E = FE$.

4. $P^{\#P_1} = P$.

5. *For every language L accepted by a logspace-uniform depth 2 AND-OR circuit family of bottom fan-in 2, $census_L$ is in FP.*

Proof. To show that (1) implies (2), let f be any function in $\#P_1$. Let M be some tally NP machine with $acc_M = f$. Assume that M runs in time n^k, for some constant k. Define

$$A \stackrel{\mathrm{df}}{=} \{x \mid |x| = n^k \text{ for some } n \text{ and } x \text{ encodes an accepting path of } M(1^n)\}.$$

Clearly, A is in P (note that n can be found in polynomial time, since computing the kth root of some integer can be done in polynomial time). Now from our hypothesis it follows that $census_A$ is in FP, and since $census_A = acc_M$, we have $f \in FP$.

Conversely, let A be an arbitrary set in P. Define M to be the tally NP machine that, on input 1^n, guesses an $x \in \{0,1\}^n$, and for each x guessed, accepts along the path for x if and only if $x \in A$. Then, $acc_M = census_A$. Since by hypothesis $acc_M \in FP$, it follows that $census_A \in FP$.

The equivalence of (2) and (3) can be proven by means of standard translation—this is essentially the function analog of Book's result that every tally NP set is in P if and only if $NE = E$ [Boo74] (see [Har83,HIS85] for the extension of this result to sparse sets).

The equivalence of (2) and (4) is straightforward.

It is easy to see that (2) implies (5). In order to prove that (5) implies (2), note that computing the number of satisfying assignments for monotone 2CNF formulas is complete for $\#P$ [Val79b] under logspace reductions. Now, given a function f in $\#P_1$, there exist logspace computable functions R, S, ρ such that for all n, $R(1^n)$ is a monotone 2CNF formula with $\rho(1^n)$ variables, and $f(1^n)$ equals the number of satisfying assignments for $R(1^n)$ divided by $S(1^n)$. The reduction R can be modified so that for every n, $\rho(1^{n+1}) > \rho(1^n)$. Now let C_m be the circuit defined as follows: (a) if $m = \rho(1^n)$ for some n, then C_m is a depth 2 AND-OR circuit that tests whether an assignment, given as the input, satisfies $R(1^n)$, and (b) if not, C_m is a depth 1 AND circuit that rejects all inputs. This circuit family $F = \{C_m\}$ is logspace-uniform. Now let A be the language accepted by F. Then, for every n, $f(1^n) = census_A(1^{\rho(1^n)})/S(1^n)$. Thus, (5) implies that $f \in FP$. ∎

Theorem 3.4 can as well be stated for more general classes than $\#P_1 = \#_1 \cdot P$. In particular, this comment applies to $\#_1 \cdot \mathcal{C}$, where for instance $\mathcal{C} = NP$ or $\mathcal{C} = PH$. Noticing that $spanP_1 = \#_1 \cdot NP$ and focusing on the first two conditions of Theorem 3.4, this observation is exemplified as follows.

Theorem 3.5   1. *Every NP set has an FP-computable census function if and only if $spanP_1 \subseteq FP$.*

2. *Every set in PH has an FP-computable census function if and only if $\#_1 \cdot PH \subseteq FP$.*

9

We will show later that the conditions of Theorem 3.4 in fact are equivalent to the two conditions stated in either part of Theorem 3.5. Next, we give some more evidence that the collapse $\#P_1 \subseteq FP$ is unlikely to hold.

Theorem 3.6 *If $\#P_1 \subseteq FP$, then the following holds:*

1. *For any fixed $k \geq 2$, $PH \subseteq MOD_kP$, and*

2. *$P = BPP$.*

Proof. For the first part, notice that Toda and Ogihara [TO92] show that for each $k \geq 2$ and any set L, if $L \in PH$, then $L \in MOD_kP/poly$ with an advice computable in (the function analog of the language class) PH^{MOD_kP}. Also, they show that for every $k \geq 2$, $PH^{MOD_kP} \subseteq P^{\#P[1]}$, where the [1] in the superscript indicates that on every input at most one call to the $\#P$ oracle is allowed. Thus, the advice function for L is in $FP_1^{\#P[1]}$. Fix $k \geq 2$ and $L \in PH$, and take an advice function $f \in FP_1^{\#P[1]}$ that puts L into $MOD_kP/poly$. Let T be the polynomial-time oracle transducer with function oracle $g \in \#P$ that witnesses $f \in FP_1^{\#P[1]}$. W.l.o.g., assume that T makes exactly one oracle call on each input (by asking a dummy query if necessary). Define the $\#P_1$ function g_1 that, on input 1^n, returns the value $g(q_n)$, where q_n is the one query string computed by T on input 1^n. Thus, f in fact is computable in $FP_1^{\#P_1[1]}$ and so, by our supposition, in polynomial time. Since L is in $MOD_kP/poly$ with polynomial-time computable advice, it follows that $L \in MOD_kP$. Hence, $PH \subseteq MOD_kP$.

In order to prove the second part, notice that BPP is in $P/poly$ [Adl78] with an advice computable in (the function analog of) PH [Sip83,Lau83], and that $PH \subseteq P^{\#P[1]}$ by Toda's Theorem [Tod91]. An argument similar to the above shows that $P = BPP$. ∎

Now we show that the conditions of Theorem 3.4 in fact are equivalent to the two conditions stated in either part of Theorem 3.5. To this end, we establish the following theorem, which is interesting in its own right. Theorem 3.7 is the main technical contribution in this section.

Theorem 3.7 $\#P_1^{PH} \subseteq FP^{\#P_1^{\#P_1}}$.

Remark 3.8   1. Note that Toda's result $PH \subseteq P^{\#P[1]}$ [Tod91] immediately gives that $\#P^{PH} \subseteq \#P^{\#P[1]}$ and $\#P_1^{PH} \subseteq \#P_1^{\#P[1]}$. Observe that the oracle is a $\#P$ function. In contrast to the inclusion $\#P_1^{PH} \subseteq \#P_1^{\#P[1]}$, Theorem 3.7 establishes containment of $\#P_1^{PH}$ in a class in which only $\#P_1$ oracles occur. Though our proof also applies the techniques of [Tod91,TO92], the result we obtain seems to be incomparable with the above-mentioned immediate consequence of Toda's Theorem.

2. It is unlikely that Theorem 3.7 can be extended to FP^{PH} or even $\#P^{PH}$ being contained in $FP^{\#P_1^{\#P_1}}$, since this would imply that $FP^{PH} \subseteq FP/poly$ and thus, in particular,

would collapse the polynomial hierarchy. In contrast, the inclusion $\mathrm{FP}_1^{\mathrm{PH}} \subseteq \mathrm{FP}_1/\mathrm{poly}$ that does follow from (the proof of) Theorem 3.7 merely implies that all *tally* sets in PH have polynomial-size circuits, a true statement that has no unlikely consequences.[3]

3. The proof of Theorem 3.7 in fact establishes a more general claim. Since $\oplus\mathrm{P}^{\mathrm{PH}}/\mathrm{poly} = \oplus\mathrm{P}/\mathrm{poly}$ [TO92], Theorem 3.7 and its corollaries can be stated even with PH replaced by $\oplus\mathrm{P}^{\mathrm{PH}}$ (note that $\oplus\mathrm{P}^{\mathrm{PH}} = \mathrm{BPP}^{\oplus\mathrm{P}}$ by Toda's result [Tod91]). However, we focus on the PH case, as this is a more natural and more central class.

Proof of Theorem 3.7. Let f be any function in $\#\mathrm{P}_1^{\mathrm{PH}}$. Note that $\#\mathrm{P}_1^{\mathrm{PH}} = \#_1 \cdot \mathrm{PH}$, since PH is closed under Turing reductions. Thus, there exist a set $L \in \mathrm{PH}$ and a polynomial p such that for each length n, $f(1^n) = |\{y \in \{0,1\}^{p(n)} \mid 1^n \#y \in L\}|$, where for convenience we assume that $p(n)$ is a power of 2 for each n. By Toda and Ogihara's result that $\mathrm{PH} \subseteq \oplus\mathrm{P}/\mathrm{poly}$ [TO92], there exist a set $A \in \oplus\mathrm{P}$, an advice function $h : \Sigma^* \to \Sigma^*$, and a polynomial q such that for each length m and each x of length m, $|h(1^m)| = q(m)$, and $x \in L$ if and only if $\langle x, h(1^m) \rangle \in A$. Let M be a machine witnessing that $A \in \oplus\mathrm{P}$, i.e., for every string z, $z \in A$ if and only if $\mathrm{acc}_M(z)$ is odd.

Toda [Tod91] defined inductively the following sequence of polynomials: For $j \in \mathbb{N}$, let $s_0(j) \overset{\mathrm{df}}{=} j$, and for each $j \in \mathbb{N}$ and $i > 0$, let

$$s_i(j) \overset{\mathrm{df}}{=} 3(s_{i-1}(j))^4 + 4(s_{i-1}(j))^3.$$

One very useful property of this sequence of polynomials is that for all $i, j \in \mathbb{N}$, $s_i(j) = c \cdot 2^{2^i}$ for some $c \in \mathbb{N}$ if j is even, and $s_i(j) = d \cdot 2^{2^i} - 1$ for some $d \in \mathbb{N}$ if j is odd (see [Tod91] for the induction proof).

We describe a polynomial-time oracle transducer T that, on input 1^n, invokes its $\#\mathrm{P}_1^{\#\mathrm{P}_1}$ function oracle g and then prints in binary the number $f(1^n)$. Fix the input 1^n. First, T transfers the input to the oracle g. Formally, function g is defined by

$$g(1^n) \overset{\mathrm{df}}{=} \sum_{y \in \{0,1\}^{p(n)}} \left(s_{\ell_n}(\mathrm{acc}_M(\langle 1^n \#y, h(1^{n+1+p(n)}) \rangle)) \right)^2,$$

where $\ell_n \overset{\mathrm{df}}{=} \log p(n)$.

Informally speaking, that g is in $\#\mathrm{P}_1^{\#\mathrm{P}_1}$ follows from the properties of the Toda polynomials, from the closure of $\#\mathrm{P}$ under addition and multiplication, and from the fact that advice function h is computable in $\mathrm{FP}_1^{\#\mathrm{P}_1[1]}$. More formally, to show that $g \in \#\mathrm{P}_1^{\#\mathrm{P}_1}$, we describe a tally NP oracle machine G and a $\#\mathrm{P}_1$ oracle g_1 for G such that, for every n, the number of accepting paths of G on input 1^n with oracle g_1 equals $g(1^n)$. On input 1^n, G first gets the advice string $a_n = h(1^{n+1+p(n)})$ of length $q(n+1+p(n))$ via one call to some appropriate $\#\mathrm{P}_1$ oracle, say g_1. This is possible by the argument given in the proof

---

[3]Indeed, P/poly is known to contain *all* tally sets and even the Turing closure of the sparse sets.

of Theorem 3.6, where g_1 is described. Then, G guesses all strings y of length $p(n)$ and for each y guessed proceeds as follows. For fixed y, let j_y be a shorthand for $\mathrm{acc}_M(\langle 1^n \# y, a_n \rangle)$. Then, $(s_{\ell_n}(j_y))^2$ is a polynomial of degree $2^{2\ell_n+1}$, which is polynomial in n. Also, the coefficients of this polynomial are deterministically computable in time polynomial in n (see [Tod91]). Since $\mathrm{acc}_M \in \#\mathrm{P}$ and $\#\mathrm{P}$ is closed under addition and multiplication, the function mapping $\langle 1^n \# y, a_n \rangle$ to $(s_{\ell_n}(j_y))^2$ is in $\#\mathrm{P}$. Let $\tilde{G}$ be an NP machine witnessing that this function is in $\#\mathrm{P}$. Then, G on input 1^n can for each guessed y produce exactly $(s_{\ell_n}(j_y))^2$ accepting paths by simulating $\tilde{G}$ on input $\langle 1^n \# y, a_n \rangle$. Again using the closure of $\#\mathrm{P}$ under addition, it follows that $g \in \#\mathrm{P}_1^{\#\mathrm{P}_1}$, as claimed.

By the above properties of the Toda polynomials, it follows that for each y of length $p(n)$, if j_y is even, then $s_{\ell_n}(j_y) = c \cdot 2^{2^{\ell_n}}$ for some $c \in \mathbb{N}$, and if j_y is odd, then $s_{\ell_n}(j_y) = d \cdot 2^{2^{\ell_n}} - 1$ for some $d \in \mathbb{N}$. Thus, recalling that $2^{\ell_n} = p(n)$, we have

$$j_y \text{ is even} \implies (s_{\ell_n}(j_y))^2 = (c^2 \cdot 2^{p(n)-1})2^{p(n)+1}, \text{ and}$$
$$j_y \text{ is odd} \implies (s_{\ell_n}(j_y))^2 = (d^2 \cdot 2^{p(n)-1} - d)2^{p(n)+1} + 1.$$

Defining the integer-valued functions $\widehat{c}(n) \stackrel{\mathrm{df}}{=} c^2 \cdot 2^{p(n)-1}$ and $\widehat{d}(n) \stackrel{\mathrm{df}}{=} d^2 \cdot 2^{p(n)-1} - d$, we obtain

$$(s_{\ell_n}(j_y))^2 = \begin{cases} \widehat{c}(n) \cdot 2^{p(n)+1} & \text{if } j_y \text{ is even} \\ \widehat{d}(n) \cdot 2^{p(n)+1} + 1 & \text{if } j_y \text{ is odd.} \end{cases}$$

Thus, since $f(1^n) \leq 2^{p(n)}$ and since j_y is odd if and only if $1^n \# y \in L$, the rightmost $p(n)+1$ bits of the binary representation of $g(1^n)$ represent the value of $f(1^n)$. Hence, after the value $g(1^n)$ has been returned by the oracle, T can output $f(1^n)$ by printing the $p(n)+1$ rightmost bits of $g(1^n)$. This completes the proof. ∎

Since $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ implies $\mathrm{FP}^{\#\mathrm{P}_1^{\#\mathrm{P}_1}} \subseteq \mathrm{FP}$, we have from Theorem 3.7 the following corollary.

Corollary 3.9 $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ *if and only if* $\#\mathrm{P}_1^{\mathrm{PH}} \subseteq \mathrm{FP}$, *and in particular,* $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ *if and only if* $\mathrm{spanP}_1 \subseteq \mathrm{FP}$.

Corollary 3.9 together with the equivalences of Theorems 3.4 and 3.5 gives the following.

Corollary 3.10 *Every* P *set has an easy census function if and only if every set in* PH *has an easy census function.*

Köbler et al. [KST89] proved that $\mathrm{spanP} = \#\mathrm{P}$ if and only if $\mathrm{NP} = \mathrm{UP}$. Their proof also establishes the analogous result for tally sets:

Lemma 3.11 (implicit in [KST89]) $\mathrm{spanP}_1 = \#\mathrm{P}_1$ *if and only if every tally* NP *set is in* UP.

Using Lemma 3.11, we show that spanP_1 and $\#\text{P}_1$ are different classes unless $\text{NE} = \text{UE}$, or unless every sparse set in NP is low for SPP. A set S is said to be $\mathcal{C}$-*low* for some class $\mathcal{C}$ if $\mathcal{C}^S = \mathcal{C}$ (see, e.g., [Sch83,KS85,Sch87,KSTT92] for a number of important lowness results). In particular, it is known that every sparse NP set is low for P^{NP} [KS85] and for PP [KSTT92], but it is not known whether all sparse NP sets are low for SPP. Torán's result that in some relativized world there exists some sparse NP set that is not contained in $\oplus\text{P}$ [Tor88], and thus not in SPP, may be taken as some evidence that not all sparse NP sets are SPP-low. Since Corollary 3.12 relativizes, $\text{spanP}_1 \neq \#\text{P}_1$ holds relative to the same oracle.

Corollary 3.12 *If* $\text{spanP}_1 = \#\text{P}_1$, *then*

1. $\text{NE} = \text{UE}$ *and*

2. *every sparse* NP *set is low for* SPP.

Proof. The first part follows from a standard upward translation argument (as mentioned in the proof of Theorem 3.4).

For the second part, assume $\text{spanP}_1 = \#\text{P}_1$, and let S be any sparse set in NP. Clearly, S polynomial-time truth-table reduces to some tally NP set T. By Lemma 3.11, our assumption implies that $T \in \text{UP}$, and thus $T \in \text{SPP}$. Since $\text{P}^{\text{SPP}} = \text{SPP}$, $S \in \text{SPP}$. The result now follows from the self-lowness of SPP [FFK94]. ∎

4   Enumerative Approximation of Census Functions

Cai and Hemaspaandra [CH89] introduced the notion of enumerative counting as a way of approximating the value of a $\#\text{P}$ function deterministically in polynomial time.

Definition 4.1 [CH89]   *Let* $f : \Sigma^* \to \Sigma^*$ *and* $g : \mathbb{N} \to \mathbb{N}$ *be two functions. A Turing transducer* E *is a* $g(n)$-*enumerator of* f *if for all* $n \in \mathbb{N}$ *and* $x \in \Sigma^n$,

1. E *on input* x *prints a list* $\mathcal{L}_x$ *with at most* $g(n)$ *elements, and*

2. $f(x)$ *is a member of list* $\mathcal{L}_x$.

A function f *is* $g(n)$-*enumerable in time* $t(n)$ *if there exists a* $g(n)$-*enumerator of* f *that runs in time* $t(n)$.

A set is $g(n)$-*enumeratively rankable in time* $t(n)$ *if its ranking function is* $g(n)$-*enumerable in time* $t(n)$.

Recall from the introduction Hemaspaandra and Rudich's result that every P set is k-enumeratively rankable for some fixed k (and indeed, even $\mathcal{O}(n^{1/2-\epsilon})$-enumeratively rankable for some $\epsilon > 0$) in polynomial time if and only if $\#\text{P} = \text{FP}$ [HR90]. They conclude that it is no more likely that one can enumeratively rank all sets in P than that one can exactly compute their ranking functions in polynomial time. We similarly characterize the question

of whether the census function of all P sets is n^α-enumerable in time n^β for fixed constants α and β. By the argument given in the proof of Theorem 3.4, this is equivalent to asking whether every $\#P_1$ function is n^α-enumerable in time n^β. We show that this implies $\#P_1 \subseteq FP$, and we thus conclude that it is no more likely that one can n^α-enumerate the census function of every P set in time n^β than that one can precisely compute its census function in polynomial time. It would be interesting to know if this result can be improved to hold for polynomial time instead of time t for some fixed polynomial $t(n) = n^\beta$.

Theorem 4.2 *Let $\alpha, \beta > 0$ be constants. If every $\#P_1$ function is n^α-enumerable in time n^β, then $\#P_1 \subseteq FP$.*

Proof. Cai and Hemaspaandra [CH91] show that for any fixed k, if $\#$SAT (the function mapping any boolean formula f to the number of satisfying assignments of f) is n^k-enumerable, then $\#P \subseteq FP$. In order to prove this, they develop the following protocol for computing the permanent of an $m \times m$ matrix A, given as parameters (the encoding of) a polynomial-time transducer E (the enumerator for $\#$SAT), and a prime number p: Set $A_0 = A$ to the input matrix and repeat the following steps for $i = 1, \ldots, m - 1$:

1. Construct from A_{i-1} an $(m - i) \times (m - i)$ matrix $B_i(X)$, defined by

$$B_i(X) \stackrel{\text{df}}{=} \sum_{k=1}^{m-i} e_k(X) a_{1k} A_{i-1}^{(1,k)},$$

   where $e_k(X)$ is a degree $(m - i)$ polynomial in X such that $e_k(X) \equiv 1$ if $X = k$ and 0 otherwise, a_{1k} is the $(1, k)$ entry of A_{i-1}, and $A_{i-1}^{(1,k)}$ is the $(1, k)$-minor of A_{i-1}. Each matrix is viewed as a matrix over $\mathbb{Z}/p\mathbb{Z}$, that is, the matrix entries are reduced modulo p. Then the following conditions hold.

   - Each entry of $B_i(X)$ is a degree $(m - i)$ polynomial in X with coefficients in $\{0, \ldots, p - 1\}$, so $\text{perm}(B_i(X))$ is a degree $(m - i)^2$ polynomial in X.
   - $\sum_{k=1}^{m-i} \text{perm}(B_i(k)) = \text{perm}(A_{i-1})$.

2. Encode $B_i(X)$ into a binary string specifying in binary p, m, and the coefficients of $B_i(X)$. There is some fixed constant $c > 0$ such that the encoding length is at most $c(m - i)^3 \log p$. Define $Q_i(X) \stackrel{\text{df}}{=} \text{perm}(B_i(X))$. Then, Q_i is a polynomial of degree at most $(m - i)^2$, whose coefficients are each length-bounded by a fixed polynomial in p and m. Thus, there is a $\#P$ function G that maps $B_i(X)$ to a number from which the coefficients of Q_i can be decoded in polynomial time.

3. Use E as an enumerator for G to obtain candidates $g_1, \ldots, g_t$. These are all degree $(m - i)^2$ polynomials that are pairwise distinct. Since two distinct degree $(m - i)^2$ polynomials can agree at no more than $(m - i)^2 - 1$ points, there are fewer than $t^2(m - i)^2 \leq t^2 m^2 - 1$ points X at which any two candidate polynomials agree. Thus,

14

if $p \geq t^2 m^2$, then there is an $r \in \{0, \ldots, p-1\}$ such that $g_j(r) \neq g_k(r)$ for all $j \neq k$. Take the smallest such r and set A_i to $B_i(r)$ with the entries reduced modulo p. Now, $\mathrm{perm}(A_i)$ modulo p specifies which g_j is correct, so we can recover $\mathrm{perm}(A_{i-1})$ modulo p in polynomial time.

At the end of this loop, A_m is a 1×1 matrix, so its permanent is easy to compute. Now working backwards again, we can recover $\mathrm{perm}(A)$ modulo p. If we do this for polynomially (in the encoding length of A) many distinct primes, then by the Chinese Remainder Theorem, we can recover the exact value of $\mathrm{perm}(A)$.

Valiant [Val79a] showed that the permanent of matrices whose entries are from the set $\{-1, 0, 1, 2\}$ is complete for #P. Analogously, we can show that there exists an infinite sequence of matrices $[M_1, M_2, \ldots]$ such that (i) the mapping $1^n \to \mathrm{perm}(M_n)$ is complete for $\#\mathrm{P}_1$, (ii) the mapping $1^n \to M_n$ is polynomial-time computable, and (iii) for every n, M_n is an $n \times n$ matrix whose entries are from $\{-1, 0, 1, 2\}$. Because of (iii), $\mathrm{perm}(M_n) \leq 2^{2n}$ for all n. So, by the Chinese Remainder Theorem, for every n, the exact value of $\mathrm{perm}(M_n)$ can be computed from $\mathrm{perm}(M_n)$ modulo p for $2n$ arbitrary distinct primes p. Define polynomials q and s by $q(n) = \langle n, n, n, 2n \rangle$ and $s(n) = q(n)^{2\alpha} n^2$. Define the function f from the tally strings to the set of natural numbers as follows.

- If $m = \langle H, n, i, j \rangle$ for some H, $i \leq n$ and $j \leq 2n$, then $f(1^m)$ is $G(B_i(X))$ defined in the above protocol when we simulate the protocol under the following constraints:

  - The jth smallest prime $> s(n)$ is used in place of p.

  - M_n is used in place of the input matrix A_0.

  - H is viewed as (the encoding of) a Turing transducer and is used in place of the enumerator E. Here, for each k with $1 \leq k \leq i-1$, the input given to H in the kth round of the protocol is $\langle H, n, k, j \rangle$, not the matrix A_k. Also, H is supposed to run in $q(n)^\beta$ steps and generates at most $q(n)^\alpha$ candidates in each round. If H does not halt in $q(n)^\beta$ steps or generates more than $q(n)^\alpha$ candidates at any point of the simulation, then the simulation is immediately aborted and the value $f(1^m)$ is set to 0.

- If m is not of the above form, $f(1^m)$ is 0.

This function f is in $\#\mathrm{P}_1$. First, there are only $i \leq m$ rounds to be simulated and each round requires m^α steps for candidate generation and some polynomial (in n) number of steps for other computations. Second, by the Prime Number Theorem, the $2n$th smallest prime $> n$ is $\mathcal{O}(n)$, so finding the jth smallest prime $> s(n)$ requires only a polynomial number of steps.

Now, by our assumption, there is an m^α-enumerator $\widehat{E}$ for f that runs in time m^β. Since the number of candidates that $\widehat{E}$ generates is at most m^α and the dimension of the matrix M_n is n, we have a prime $> m^{2\alpha} n^2$. This implies that with $\widehat{E}$ as the enumerator, for every $n \geq \widehat{E}$, every $j, 1 \leq j \leq 2n$, and every $i, 1 \leq i \leq n$, we successfully find an r for

15

distinguishing the candidates. So, with $\widehat{E}$ as the enumerator, for all $n \geq \widehat{E}$, $\mathrm{perm}(M_n)$ is polynomial-time computable. Hence $\#\mathrm{P}_1 \subseteq \mathrm{FP}$. ∎

5 Oracle Results

In this section, we provide a number of relativized results on the existence or non-existence of P sets simultaneously satisfying pairs of conditions chosen among the properties (i), (ii), and (iii) from Section 3. For instance, Theorem 5.1 and its Corollary 5.2 below exhibit a relativized world in which every P set has an easy census function (Property (ii)), yet there exists some set in P that is not rankable (Property (i)).

Theorem 5.1 *There exists an oracle D such that $\#\mathrm{P}_1^D \subseteq \mathrm{FP}^D \neq \#\mathrm{P}^D$.*

From the relativized versions of Theorem 3.4 and of Hemaspaandra and Rudich's result in [HR90] that every P set is rankable if and only if $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}$ (which is equivalent with $\mathrm{FP} = \#\mathrm{P}$, and this equivalence itself also relativizes), we immediately obtain the following corollary.

Corollary 5.2 *There exists an oracle D such that all sets in P^D have a census function computable in FP^D, yet there exists some set in P^D that is not rankable by any function in FP^D.*

Proof of Theorem 5.1. Balcázar et al. [BBS86] and Long and Selman [LS86] proved that the polynomial hierarchy does not collapse if and only if it does not collapse relative to every sparse oracle. Since their proof relativizes (i.e., it applies to the relativized polynomial hierarchy as well), we have the following claim:

Claim 5.3 [BBS86,LS86] *For every set B, PH^B does not collapse if and only if for every sparse oracle S, $(\mathrm{PH}^B)^S$ does not collapse.*

Note that $(\mathrm{PH}^B)^S = \mathrm{PH}^{B \oplus S}$, where $X \oplus Y \overset{\mathrm{df}}{=} \{0x \mid x \in X\} \cup \{1y \mid y \in Y\}$ denotes the join of any two sets X and Y. Fix an oracle A such that PH^A does not collapse (such oracles were constructed by Yao [Yao85], Håstad [Hås89], and Ko [Ko89] who built on the work of Furst et al. [FSS84]). Then, by Claim 5.3 above, for every sparse set S, $\mathrm{PH}^{A \oplus S}$ does not collapse. So, in particular, $\mathrm{P}^{A \oplus S} \neq \mathrm{NP}^{A \oplus S}$ for every sparse set S. Since for every oracle B, $\#\mathrm{P}^B = \mathrm{FP}^B$ implies $\mathrm{NP}^B = \mathrm{P}^B$, we have that $\#\mathrm{P}^{A \oplus S} \neq \mathrm{FP}^{A \oplus S}$ for every sparse set S.

So it remains to prove that there exists a sparse set T such that $\#\mathrm{P}_1^{A \oplus T} \subseteq \mathrm{FP}^{A \oplus T}$. Then, setting $D = A \oplus T$ completes the proof.

Assume that our pairing function $\langle \cdot, \cdot, \cdot \rangle$ is nondecreasing in each parameter, polynomial-time computable and invertible, and is one-to-one and onto. Let $N_1^{(\cdot)}, N_2^{(\cdot)}, \ldots$ be a standard

enumeration of all tally NP oracle machines. For each $i \geq 1$, let p_i be the polynomial time bound of $N_i^{(\cdot)}$. Then, the function $f^{(\cdot)}$ defined by

$$f^{(\cdot)}(1^{\langle i,n,j \rangle}) \quad \overset{\mathrm{df}}{=} \quad \begin{cases} \mathrm{acc}_{N_i^{(\cdot)}}(1^n) & \text{if } p_i(n) < j \\ 0 & \text{otherwise} \end{cases}$$

is a canonical function complete for the class $\#\mathrm{P}_1^{(\cdot)}$.[4] In particular, for every fixed set S, $f^{(A \oplus S)}$ is complete for $\#\mathrm{P}_1^{A \oplus S}$.

The oracle set T is defined in such a way that, for any given $m = \langle i, n, j \rangle$ in unary, some polynomial-time oracle transducer can retrieve the value of $f^{(A \oplus T)}(1^m)$ from its oracle $A \oplus T$ by asking at most m queries. More formally, we construct T in stages such that for each $m = \langle i, n, j \rangle$:

$$1^k 0^{m-k} \# b \in T \iff 1 \leq k \leq |f^{(A \oplus T)}(1^m)| \text{ and the } k\text{th bit of } f^{(A \oplus T)}(1^m) \text{ is } b.$$

Since by the above definition, $|f^{(A \oplus T)}(1^m)| < m$ and so, in particular, $N_i^{A \oplus T}(1^n)$ cannot query strings of length $\geq m$, there is no interference between the stages of the construction of T. It is easy to see that T is a sparse set satisfying $\#\mathrm{P}_1^{A \oplus T} \subseteq \mathrm{FP}^{A \oplus T}$. ∎

Now we construct an oracle relative to which there exists some scalable set in P whose census function is not easy to compute.

Theorem 5.4 *There exists an oracle A such that there exists an A-scalable set B whose census function is not in FP^A.*

Proof. We will construct A and B in such a way that B is P^A-isomorphic to the set $R \overset{\mathrm{df}}{=} \{0x \mid x \in \Sigma^*\}$, which is rankable in FP (and thus in FP^A). For each $n \geq 1$, we have $census_R(1^n) = 2^{n-1}$. So $census_R$ is easy to compute, but we want B to have a hard census function. In light of Proposition 3.1.2, we thus need the isomorphism, f, between B and R be non-length-preserving. In particular, we will define f so as to satisfy $|f(x)| \leq |x| + 1$ and $|f^{-1}(y)| \leq |y|$ for all $x, y \in \Sigma^*$. When f is defined, we let B be the set $f^{-1}(R)$. To have f and its inverse computable in FP^A, we encode f and f^{-1} into $A \overset{\mathrm{df}}{=} A_f \oplus A_{f^{-1}}$ as follows. For all $x \in \Sigma^*$, $i \geq 1$, and $b \in \{0, 1\}$, we ensure that

(1) $$\langle x, i, b \rangle \in A_{f^*} \iff \text{the } i\text{th bit of } f^*(x) \text{ is } b,$$

where f^* stands for either f or f^{-1}. At the same time we diagonalize against FP^A so as to ensure $census_B \notin \mathrm{FP}^A$.

Let $T_1^{(\cdot)}, T_2^{(\cdot)}, \ldots$ be a standard enumeration of all deterministic polynomial-time oracle transducers, and let $p_1, p_2, \ldots$ be a sequence of strictly increasing polynomials such that p_i bounds the running time of T_i (independent of the oracle used). By (1) above, implicit in

---

[4]See [Val79b] for *natural* $\#\mathrm{P}_1$-complete functions.

the definition of f and f^{-1} is the definition of A, so it suffices to construct the isomorphism. The construction of f and f^{-1} is in stages. By the end of stage i, f will have been defined for all strings of length up to $r(i)$, where r will be determined below. Initially, we start with $r(0) = 0$, and we define $f(\epsilon) = \epsilon$. Stage $i > 0$ of the construction is as follows.

Stage i: Choose n_i to be the smallest integer such that $n_i > r(i-1)$ and $p_i(n_i) < 2^{n_i-2}$. Let A' be the subset of A that has been decided by now. We want to define f so that, eventually, $T_i^A(1^{n_i}) \neq census_B(1^{n_i})$. Simulate $T_i^{A'}$ on input 1^{n_i}. Whenever in this simulation a string of the form $0\langle x, i, b\rangle$ whose membership in A has not yet been decided is queried, we add this string to A' and set the ith bit of $f(x)$ to b unless we have already put $0\langle x, i, 1-b\rangle$ into A (and thus have set this bit to $1 - b$), or unless $i > |x| + 1$. The same comment applies to query strings $1\langle y, j, b\rangle$ whose membership in A has not been decided yet and which may fix the jth bit of $f^{-1}(y)$. If we added the queried string to A', we continue the simulation in the "yes" state; otherwise, in the "no" state. In this way, the simulation of $T_i^{A'}(1^{n_i})$ may determine f (and f^{-1}) on at most $p_i(n_i) < 2^{n_i-2}$ bits of the strings of length n_i. Thus, for no $m \geq n_i$ is f^{-1} determined on all strings of length m in R or $\overline{R}$. Once the value $T_i^{A'}(1^{n_i})$ is computed, there is room to decide $f(x)$ and $f^{-1}(y)$ for all strings x and y of lengths between $r(i-1)$ and $p_i(n_i)$ so that f is an isomorphism mapping to $\bigcup_{\ell=r(i-1)}^{p_i(n_i)} R^{=\ell}$ and such that $census_B(1^{n_i}) \neq T_i^{A'}(1^{n_i})$, without changing the output value of $T_i^{A'}(1^{n_i})$. Finally, define $r(i) = p_i(n_i)$. ∎

Next, we provide an oracle relative to which there exists some set in P that is neither scalable nor has an easy census function.

Theorem 5.5 *There exists an oracle D such that $D \in \mathrm{P}^D$ is not D-scalable and its census function is not in FP^D.*

Proof. This is a simple interweaving of two diagonalizations. The only question is how to construct a non-scalable set.

It is known from the work of Goldsmith and Homer [GH96] that any sparse set is scalable if and only if it is rankable, and this holds if and only if it is P-printable.[5] D will be sparse, with at most 2 strings at each length. We assume that $(T_i^{(\cdot)})_{i \geq 1}$ enumerates $\mathrm{FP}^{(\cdot)}$, and that $T_i^{(\cdot)}$ runs in time n^i.

At stage $2i$, we guarantee that $T_i^D(1^n)$ does not compute the rank of 1^n in D, where n is chosen large enough that $n^i < 2^n$. For this n, we put 1^n into D. Compute $T_i^D(1^n)$, restraining any oracle strings of length $\geq n$ that it queries. By our choice of n, this does not decide $D^{=m}$ for any $m \geq n$, so we can then put in the appropriate number of strings of length n for the diagonalization.

---

[5]A set is P-*printable* [HY84] if there exists a polynomial-time transducer T such that for each length n, T on input 1^n prints a list of all elements of the set up to length n.

At stage $2i + 1$ we guarantee that $T_i^D(1^n)$ does not compute the census function of D, where n is chosen large enough that $n^i < 2^n$. Again, compute $T_i^D(1^n)$, restraining any oracle strings of length $\geq n$ that it queries. By our choice of n, this does not decide $D^{=m}$ for any $m \geq n$, so we can then put in the appropriate number of strings of length n for the diagonalization. ∎

Finally, we show that relative to an oracle, there exists some non-scalable set in P having an easy census function.

Theorem 5.6 *There exists an oracle A such that $A \in \mathrm{P}^A$ is not A-scalable and its census function is in FP^A.*

Proof. We construct the oracle A so that A has one string of each length. For those lengths for which nothing else is decided, we put in 1^n. Otherwise, we do the following.

To make the oracle A non-A-scalable, we actually make it non-P^A-printable. At stage i, choose an appropriate length n, and then compute $T_i^A(1^n)$. Whenever it queries a string of length $\geq n$, restrain the string from the oracle. If it does anything except print out $A^{\leq n}$, then put in the first unrestrained string of each length. If it correctly prints A up to length n, then choose an x of each relevant length to include that neither is restrained nor printed. ∎

We conclude this section with a remark on a technical difficulty in proving the following statement: "There exists an oracle E such that all sets in P^E have a census function computable in FP^E, but $E \in \mathrm{P}^E$ is not E-scalable." Call this statement (S). One might hope to prove (S) by exploiting again the fact that scalability, rankability, and P-printability are equivalent properties on the sparse sets [GH96], which was useful in the proofs of Theorems 5.5 and 5.6. Now, replacing in (S) non-scalability by non-rankability makes (S) the following stronger version of Theorem 5.1: "There exists a *sparse* set E such that $\#\mathrm{P}_1^E \subseteq \mathrm{FP}^E \neq \#\mathrm{P}^E$." However, since the oracle $D = A \oplus T$ constructed in the proof of Theorem 5.1 inherently is a *non*sparse set due to its A part (and it cannot be made sparse unless one could separate the unrelativized polynomial hierarchy [LS86,BBS86]), this approach does not work to prove (S). Therefore, to prove (S), one would need to construct a nonsparse set E with the desired properties, and we leave this as an interesting open issue.

Acknowledgments. We are deeply indebted to Lance Fortnow, Lane Hemaspaandra, and Gabriel Istrate for interesting discussions and for helpful comments and suggestions, and we thank Eric Allender and Lane Hemaspaandra for pointers to the literature.

References

[Adl78]   L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, pages 75–83, 1978.

[All91]   E. Allender. Limitations of the upward separation technique. *Mathematical Systems Theory*, 24(1):53–67, 1991.

[AR88]   E. Allender and R. Rubinstein. P-printable sets. *SIAM Journal on Computing*, 17(6):1193–1202, 1988.

[BBS86]   J. Balcázar, R. Book, and U. Schöning. The polynomial-time hierarchy and sparse oracles. *Journal of the ACM*, 33(3):603–617, 1986.

[BC93]   D. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1993.

[BG92]   R. Beigel and J. Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 103(1):3–23, 1992.

[BH77]   L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[Boo74]   R. Book. Tally languages and complexity classes. *Information and Control*, 26:186–193, 1974.

[CH89]   J. Cai and L. Hemachandra. Enumerative counting is hard. *Information and Computation*, 82(1):34–44, 1989.

[CH90]   J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 23(2):95–106, 1990.

[CH91]   J. Cai and L. Hemachandra. A note on enumerative counting. *Information Processing Letters*, 38(4):215–219, 1991.

[FFK94]   S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[FSS84]   M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[GH96]   J. Goldsmith and S. Homer. Scalability and the isomorphism problem. *Information Processing Letters*, 57:137–143, 1996.

[Gil77]   J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977.

[GP86]   L. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of boolean functions. *Theoretical Computer Science*, 43:43–58, 1986.

[GS91]   A. Goldberg and M. Sipser. Compression and ranking. *SIAM Journal on Computing*, 20(3):524–536, 1991.

[Har83]   J. Hartmanis. On sparse sets in NP−P. *Information Processing Letters*, 16:55–60, 1983.

[Hås89]   J. Håstad. Almost optimal lower bounds for small depth circuits. In S. Micali, editor, *Randomness and Computation,* volume 5 of *Advances in Computing Research*, pages 143–170. JAI Press, Greenwich, 1989.

[Hem89]   L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299–322, 1989.

[Her90]   U. Hertrampf. Relations among MOD-classes. *Theoretical Computer Science*, 74(3):325–328, 1990.

[HHH]   E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A downward collapse within the polynomial hierarchy. *SIAM Journal on Computing.* To appear.

[HIS85]   J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP−P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):159–181, 1985.

[HJ95]   L. Hemaspaandra and S. Jha. Defying upward and downward separation. *Information and Computation*, 121:1–13, 1995.

[HJRW]   L. Hemaspaandra, Z. Jiang, J. Rothe, and O. Watanabe. Boolean operations, joins, and the extended low hierarchy. *Theoretical Computer Science.* To appear.

[HR90]   L. Hemachandra and S. Rudich. On the complexity of ranking. *Journal of Computer and System Sciences*, 41(2):251–271, 1990.

[HR97]   L. Hemaspaandra and J. Rothe. Unambiguous computation: Boolean hierarchies and sparse Turing-complete sets. *SIAM Journal on Computing*, 26(3):634–653, June 1997.

[HRW97a]   L. Hemaspaandra, J. Rothe, and G. Wechsung. Easy sets and hard certificate schemes. *Acta Informatica*, 34(11):859–879, 1997.

[HRW97b]   L. Hemaspaandra, J. Rothe, and G. Wechsung. On sets with easy certificates and the existence of one-way permutations. In *Proceedings of the Third Italian Conference on Algorithms and Complexity*, pages 264–275. Springer-Verlag *Lecture Notes in Computer Science #1203*, March 1997.

[HU79]   J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[HY84]   J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoretical Computer Science*, 34:17–32, 1984.

[KL80]   R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, pages 302–309, April 1980. An extended version has also appeared as: Turing machines that take advice, *L'Enseignement Mathématique*, 2nd series 28, 1982, pages 191–209.

[Ko89]   K. Ko. Relativized polynomial time hierarchies having exactly k levels. *SIAM Journal on Computing*, 18(2):392–408, 1989.

[KS85]   K. Ko and U. Schöning. On circuit-size complexity and the low hierarchy in NP. *SIAM Journal on Computing*, 14(1):41–51, 1985.

[KST89]   J. Köbler, U. Schöning, and J. Torán. On counting and approximation. *Acta Informatica*, 26:363–379, 1989.

[KSTT92] J. Köbler, U. Schöning, S. Toda, and J. Torán. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44(2):272–286, 1992.

[Lau83]   C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 14:215–217, 1983.

[Lon85]   T. Long. On restricting the size of oracles compared with restricting access to oracles. *SIAM Journal on Computing*, 14(3):585–597, 1985. Erratum appears in the same journal, 17(3):628.

[LS86]   T. Long and A. Selman. Relativizing complexity classes with sparse oracles. *Journal of the ACM*, 33(3):618–627, 1986.

[Mah82]   S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.

[MS72]   A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.

[OH93]   M. Ogiwara and L. Hemachandra. A complexity theory for closure properties. *Journal of Computer and System Sciences*, 46(3):295–325, 1993.

[Pap94]   C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[PZ83]   C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings 6th GI Conference on Theoretical Computer Science*, pages 269–276. Springer-Verlag *Lecture Notes in Computer Science #145*, 1983.

[RRW94]   R. Rao, J. Rothe, and O. Watanabe. Upward separation for FewP and related classes. *Information Processing Letters*, 52:175–180, 1994.

[Sch83]   U. Schöning. A low and a high hierarchy within NP. *Journal of Computer and System Sciences*, 27:14–28, 1983.

[Sch87]   U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, 1987.

[Sip83]   M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.

[Sto77]   L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[TO92]   S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2):316–328, 1992.

[Tod91]   S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Tor88]   J. Torán. *Structural Properties of the Counting Hierarchies*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 1988.

[Val76]   L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5:20–23, 1976.

[Val79a]   L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Val79b]   L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[Wel93]   D. Welsh. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, 1993.

[Yao85]   A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.